1 Looped Pipeline Architecture

Many quantum computing platforms are based on a two-dimensional physical layout. We focus on ”looped pipelines,” [1] which offer the advantages of a three-dimensional lattice while being restricted to two-dimensional space. This architecture leverages qubit shuttling, where qubits are moved around on the chip, enabling long-range interactions between qubits that are far apart.

1.1 Classical Linear Pipeline

./pdf/classical_pipeline-eps-converted-to.pdf
Fig. 1

Let us fist introduce ”data-processing pipeline,” we divide data-processing circuit into multple steps with each step able to operate only one dataset at a time shown in Fig. **??**. Now intead of inputting second dataset after the processing the first dataset on whole circuit, we input it as soon as finishing the process of the first dataset on step 1, similarly all subseaquent datasets and subseaquent steps. In this way, all data-processing steps can be working on different datasets in parallel, increasing the throughput of the system. this is a concept of pipelining. The processing time of the mth step is denoted as τ_m, and suppose there exist M steps in the circuit. Then, the total time T_{circ} taken for the entire circuit to process one dataset is given by:

$$T_{\mathrm{circ}} = \sum_{m=1}^{M} \tau_m. \tag{1}$$

This is the time required for every dataset without pipelining, and it is also the time required to process the first dataset in pipelining. For the second dataset in pipelining, the additional time required for processing is given by:

$$\tau_{\max} = \max_i \tau_i. \tag{2}$$

which is called ”rate-limiting step.” From the second dataset onward, the time required for processing each dataset equals the maximum of τ_i. Hence, the total time for processing the kth datasets are given by:

$$T_{\mathrm{pipe}} = T_{\mathrm{circ}} + (k-1)\tau_{\max}. \tag{3}$$

We call this the ”steady-flow” pipelining scheme, as the data stream can flow through the entire pipeline without requiring modifications to the time gap between adjacent datasets or being put on hold at any point along the pipeline. Deviating from the steadyflow scheme, we often need to tempolary put the dataset on hold in the pipeline. For this, we need to implement ”baffer” where multiple datasets can be on hold.

1.2 Looped Qubit Pipeline

Similar to the classical pipeline, we can define the linear qubit pipeline, as shown in Fig. **??**(a). The qubits are processed at each step, which consists of shuttling, gate operations, initialization, or

./pdf/shuttling_track-eps-converted-to.pdf

Fig. 2

measurement. We can apply the steady-flow scheme to this pipeline, so the time required to process k qubits is given by Eq. **??**. Note that, in practice, the rate-limiting step is often not the shuttling step, as the shuttling operation is relatively faster compared to other operations. Without loss of generality, we assume that the shuttling operation can hold the qubits without pushing them forward, allowing it to act as a buffering region for the pipelining when needed.

Beyond the linear qubit pipeline, we can define the looped qubit pipeline, as shown in Fig. **??**. Unlike the linear qubit pipeline, where the circuits that can be performed on the qubits are restricted by the number of devices on the pipeline, the looped pipeline allows the qubits to circulate around the loop and reuse the gate devices. This enables the effective execution of circuits with any depth on the qubits.